Trois tris

Question préliminaire : supposons que `L` est une liste, avec i et j deux indices distincts compris entre 1 et la longueur de `L`. Comment échanger les valeurs situées en position i et j dans `L` ?

1 Un tri par sélection

Il s'agit d'un tri itératif (basé sur des boucles). On trouve le plus gros élément, on le place, et on continue avec le second plus gros, etc...

1. Ecrire une fonction `maximum` prenant en entrée une liste et retournant le plus grand des éléments (on supposera la liste effectivement constituée d'entiers ou de réels ; votre programme n'a PAS à le vérifier).
2. Ecrire une fonction `pos_max` prenant en entrée une liste et retournant la position du plus grand des éléments. Si le maximum est présent plusieurs fois, on retournera impérativement la position la plus élevée.
3. Ecrire une fonction `pos_max_parmi` prenant en entrée une liste et un entier N et retournant la position du plus grand des éléments parmi les N premiers. On supposera que N est inférieur ou égal au nombre d'éléments de la liste.
4. Ecrire une fonction `tri_selec` prenant en entrée une liste `L` et retournant cette liste triée.
5. Lors d'une évaluation de `tri_selec(L)`, évaluer le nombre de comparaisons de type `x>L[i]` ou bien `L[i]>L[j]` qui seront effectuées, en fonction de la longueur n de la liste.

2 Un tri par insertion

Encore un tri itératif, basé sur le principe suivant : au début, la sous-liste `L[1..1]` est triée ; on va insérer dans cette liste petit à petit `L[2],L[3]`, etc... jusqu'à `L[n]`, le principe étant qu'après avoir inseré `L[I]`, le sous-tableau `L[1..I]` sera trié.

Pour insérer `L[I]`, on échange si besoin est le couple $(I-1, I)$, puis éventuellement le couple $(I-2, I-1)$, etc... jusqu'au couple $(1, 2)$. Si à un moment on n'a pas fait l'échange, cela signifie que `L[k]<=L[k+1]`, et on arrête (pour ce I là).

1. Pour quelles valeurs successives de I va-t-on devoir insérer `L[I]` dans `L[1..I-1]` ?
2. Simuler à la main l'insertion de `L[I]` dans `L[1..I-1]` lorsque `L=[2,3,6,10,12,4,9,5,16]` et $I = 6$.
3. Pour I fixé, l'insertion de `L[I]` dans `L[1..I-1]` est de la forme suivante : "tant que $k \geq 1$ et que `L[k]>L[k+1]`, faire ...". Alors, on fait quoi au juste ? Et k prend quelle valeur avant de lancer la boucle "tant que" ?

4. Ecrire une fonction `tri_inser` prenant en entrée une liste `L` et retournant cette liste triée.
5. Lors d'une évaluation de `tri_inser(L)`, évaluer le nombre de comparaisons de type `L[i]>L[j]` qui seront effectuées, en fonction de la longueur n de la liste.

3 Le tri fusion (ou dichotomique)

Il s'agit d'un tri récursif "diviser pour rêgner" : on va trier récursivement la première moitié et la seconde moitié de la liste ; ensuite, on va *fusioner* ces deux listes triées L_1 et L_2. Pour cela, on va mettre au fur et à mesure dans une suite le plus petit élément de $L_1 \cup L_2$ non encore inseré.

1. On suppose que `L` possède N éléments. Quels seront les indices de début et de fin des deux demi-listes ? Vérifier impérativement avec $N = 6$ mais aussi $N = 7$.
2. Commencer l'écritue du proramme `tri_fusion` prenant en entrée une liste, et retournant dans un premier temps les deux demi-listes. Tester dans des cas pair/impair.
3. Pour fusionner deux listes triées L_1 et L_2, de tailles respectives n_1 et n_2[1]. On va insérer les différents éléments dans une SUITE initialement vide (`s:=NULL` pour créer une telle suite). On prend deux indices i_1 et i_2 partant de 1, qui vont décrire les indices de L_1 et L_2. Tant que ($i_1 \leq n_1$ et $i_2 \leq n_2$), on regarde qui de `L_1[i_1]` ou `L_2[i_2]` est le plus grand. On place alors cet élément à la fin de `s` ; on incrémente i_1 ou i_2, et on recommence. Quand on sort de la boucle "Tant que", cela signifie qu'une des deux listes a été complètement insérée ; il reste à insérer le reste de l'autre à la fin de `s`.

   Appliquer ce principe pour écrire complètement la fonction `tri_fusion`.
4. (a) Si une liste est de longueur n, on note $C(n)$ le nombre d'"opérations élémentaires" (du type comparaison, ou affectations) effectuées lors d'un appel à `tri_fusion` sur une telle liste. Donner une formule simple reliant $C(2n)$ à $C(n)$.

   (b) En déduire la valeur de $C(2^k)$.

   (c) Maintenant, à n fixé, on peut choisir k tel que $2^k < n \leq 2^{k+1}$ et on a de façon claire $C(n) \leq C(2^{k+1})$. Prouver grâce à cela que $C(n) = O(n \ln n)$.

[1] Ca ne me dérange pas que vous preniez d'autres noms d'indices, mais ne venez pas pleurer après parce que vous êtes perdus avec vos propres indices...